Hyperproperties: Verification of Proofs

Denis L. Bueno   Michael R. Clarkson

{dlb335,clarkson}@cs.cornell.edu
Department of Computer Science
Cornell University

Computing and Information Science Technical Report
http://hdl.handle.net/1813/11153

July 25, 2008

Hyperproperties: Verification of Proofs*

Denis L. Bueno   Michael R. Clarkson
{dlb335,clarkson}@cs.cornell.edu
Department of Computer Science
Cornell University

July 25, 2008

Abstract

This paper formalizes some proofs by Clarkson and Schneider about hyperproperties. The proofs are mechanically verified using the proof assistant Isabelle.

1 Introduction

Properties are sets of execution traces, and hyperproperties are sets of properties. This paper formalizes Clarkson and Schneider's theory of hyperproperties [3] using Isabelle/HOL [4]. We present human-readable, mechanically-verified proofs of the propositions and theorems in [3]—except those related to topology, which we leave for future work. The proofs given here are formal analogues of informal proofs that were given in a previous technical report [2]. Thus, in addition to verifying the propositions and theorems, we have also verified the original proofs themselves.

This document was produced from LaTeX output, which was generated from Isabelle theory files. Those theory files are available for download from the same URL that hosts this technical report [1]. The numbering of propositions and theorems in this document follows the numbering in [2, 3].

*Supported in part by AFOSR grant F9550-06-0019, National Science Foundation Grants 0430161 and CCF-0424422 (TRUST), and a gift from Microsoft Corporation. Denis Bueno is supported by a Sandia National Laboratories Fellowship; Michael Clarkson is supported by an Intel Foundation PhD Fellowship.

theory *HyperDefs*
imports *Main LList2 LaTeXsugar OptionalSugar*
begin

notation {} ($\emptyset$)

2 Definitions

typedecl *state*
— An abstract notion of a state.

types *trace* = *state llist*
— Traces are (possibly infinite) lists of states.

consts *States* :: *state set* (Σ)
— An abstract set of states.

consts *BottomState* :: *state*
syntax (*latex*)
BottomState :: *state* ($\perp$)

consts *DummyState* :: *state*

We assume the existence of one DummyState, which is used by Theorem 3 and Proposition 3.

axioms *DummyState-is-State*: *DummyState* $\in \Sigma$

constdefs
psi-fin :: *trace set* (Ψ_{fin})
$\Psi_{\mathrm{fin}} \triangleq \Sigma^{\star}$
psi-inf :: *trace set* (Ψ_{inf})
$\Psi_{\mathrm{inf}} \triangleq \Sigma^{\omega}$
Ψ :: *trace set*
$\Psi \triangleq \Psi_{\mathrm{fin}} \cup \Psi_{\mathrm{inf}}$

types
property = *trace set*
hyperproperty = *property set*
constdefs
Prop :: *property set*

$Prop \triangleq Pow\ \Psi_{\mathrm{inf}}$

HP :: *hyperproperty set*
$HP \triangleq Pow\ Prop$

consts
property-satisfies :: *trace set* $\Rightarrow$ *property* $\Rightarrow$ *bool* ((- $\models$ -) [80,80] 80)
hyperproperty-satisfies :: *trace set* $\Rightarrow$ *hyperproperty* $\Rightarrow$ *bool* ((- $\models$ -) [80,80] 80)
defs (**overloaded**)
property-satisfies-def: $ts \models p \triangleq ts \subseteq p$
hyperproperty-satisfies-def: $ts \models h \triangleq ts \in h$

constdefs
property-lift :: *property* $\Rightarrow$ *hyperproperty* ([[-]] 80)
property-lift $p \triangleq Pow\ p$
notation *property-lift* ([-] 80)

constdefs
trace-set-prefix :: *trace set* $\Rightarrow$ *trace set* $\Rightarrow$ *bool* (**infix** $\leq$ 80)
trace-set-prefix-def:
$T \leq T' \triangleq \forall\ t.\ t \in T \longrightarrow (\exists\ t'.\ t' \in T' \wedge t \leq t')$

Obs :: *trace set set*
$Obs \triangleq \{ts.\ ts \subseteq \Psi_{\mathrm{fin}} \wedge finite\ ts\}$

sp :: *property* $\Rightarrow$ *bool*
$sp\ P \triangleq P \in Prop\ \wedge$
$(\forall\ t \in \Psi_{\mathrm{inf}}.\ t \notin P \longrightarrow$
$(\exists\ m \in \Psi_{\mathrm{fin}}.\ m \leq t\ \wedge$
$(\forall\ t' \in \Psi_{\mathrm{inf}}.\ m \leq t' \longrightarrow t' \notin P)))$
SP :: *property set*
$SP \triangleq \{P.\ sp\ P\}$

false-p :: *property*
false-p $\triangleq \emptyset$

shp :: *hyperproperty* $\Rightarrow$ *bool*
$shp\ H \triangleq H \in HP\ \wedge$
$(\forall\ T \in Prop.\ T \notin H \longrightarrow$
$(\exists\ M \in Obs.\ M \leq T\ \wedge$
$(\forall\ T' \in Prop.\ M \leq T' \longrightarrow T' \notin H)))$
SHP :: *hyperproperty set*
$SHP \triangleq \{hp.\ shp\ hp\}$

false-hp :: *hyperproperty*

false-hp $\triangleq$ [*false-p*]

lp :: *property* $\Rightarrow$ *bool*
lp $L \triangleq L \in$ *Prop* $\wedge$ ($\forall\ t \in \Psi_{\text{fin}}$. ($\exists\ t' \in \Psi_{\text{inf}}$. $t \leq t' \wedge t' \in L$))
LP :: *property set*
LP $\triangleq$ {*P*. *lp P*}
lhp :: *hyperproperty* $\Rightarrow$ *bool*
lhp $H \triangleq H \in$ *HP* $\wedge$ ($\forall\ T \in$ *Obs*. ($\exists\ T' \in$ *Prop*. $T \leq T' \wedge T' \in H$))
LHP :: *hyperproperty set*
LHP $\triangleq$ {*hp* . *lhp hp*}

true-Prop :: *property*
true-Prop $\triangleq \Psi_{\text{inf}}$
true-HP :: *hyperproperty*
true-HP $\triangleq$ *Prop*

end

theory *Hyper*
imports *HyperDefs*
begin

3 Proposition 1

3.1 Lemmas

lemma *property-lifts-into-hyperproperty*:
assumes *P-Prop*: $P \in$ *Prop*
shows [*P*] $\in$ *HP*
using *P-Prop*
unfolding *property-lift-def Prop-def HP-def* **by** *blast*

3.2 Proposition

theorem *proposition-1-oif*:
assumes *S-Prop*: $S \in$ *Prop* **and** *S-SP*: $S \in$ *SP*
shows [*S*] $\in$ *SHP*
proof –
have *lift-S-HP*: [*S*] $\in$ *HP*
using *S-Prop property-lifts-into-hyperproperty* **by** *blast*
{

fix T :: *property*
assume *T-st*: $T \in$ *Prop* $T \notin [S]$
from ‹$T \notin [S]$› **have** $\neg(T \subseteq S)$ **by** (*simp add*: *property-lift-def*)
then obtain t **where** *t-st*: $t \in T$ $t \notin S$ **by** *blast*

have $\exists\ m.\ m \in \Psi_{\text{fin}} \wedge m \leq t \wedge (\forall\ t' \in \Psi_{\text{inf}}.\ m \leq t' \longrightarrow t' \notin S)$
proof –
from *t-st* **and** *T-st* **have** *t-psi-inf*: $t \in \Psi_{\text{inf}}$
unfolding *Prop-def* **by** *blast*
with *S-Prop* **and** *S-SP* **and** *T-st* **and** *t-st*
show *?thesis* **unfolding** *SP-def Prop-def sp-def* **by** *blast*
qed
then obtain m **where** *m-st*: $m \in \Psi_{\text{fin}}$ $m \leq t$ $\forall\ t'.\ t' \in \Psi_{\text{inf}} \wedge m \leq t' \longrightarrow$
$t' \notin S$
by *blast*

let $?M = \{m\}$
from *m-st* **and** *t-st* **have** *M-prf-T*: $?M \leq T$
unfolding *trace-set-prefix-def* **by** *blast*
with *m-st* **and** *t-st* **have** *M-Obs*: $?M \in$ *Obs*
unfolding *Obs-def* **by** *blast*

{
fix T' :: *property*
assume *T′-st*: $T' \in$ *Prop* $?M \leq T'$

then have $\exists\ t' \in T'.\ m \leq t'$
by (*simp only*: *trace-set-prefix-def*) *blast*
then obtain t' **where** *t′-st*: $t' \in T'$ $m \leq t'$ **..**
with *m-st* **and** *T′-st* **have** *t′-out-S*: $t' \notin S$
unfolding *Prop-def* **by** *blast*
from *T′-st* **and** *S-Prop* **and** *S-SP* **and** *t′-st* **and** *t′-out-S*
have $T' \notin [S]$ **unfolding** *property-lift-def* **by** *blast*
}
hence $\forall\ T'.\ T' \in$ *Prop* $\wedge\ ?M \leq T' \longrightarrow T' \notin [\ S\ ]$ **by** *blast*

with *m-st* **and** *M-prf-T* **and** *M-Obs*
have $\exists\ M.\ M \in$ *Obs* $\wedge M \leq T \wedge (\forall\ T'.\ T' \in$ *Prop* $\wedge M \leq T' \longrightarrow T' \notin [S])$
by *blast*
}
thus *?thesis* **using** *lift-S-HP* **unfolding** *SHP-def shp-def* **by** *blast*
qed

lemma *prefix-set-has-longest*:

fixes t :: $'a$ *llist*
assumes *X-fin*: *finite* X **and** *X-non-empty*: $X \neq \emptyset$
and *X-prefix-t*: $\forall\ x \in X.\ x \leq t$
shows $\exists\ m \in X.\ (\forall\ x \in X.\ x \leq m)$
using *prems*
proof (*induct X rule*: *Finite-Set.finite-ne-induct*)
  fix x :: $'a$ *llist* **show** $\exists\ m \in \{x\}.\ \forall x \in \{x\}.\ x \leq m$ **by** *blast*
next
  fix x :: $'a$ *llist* **and** F::$'a$ *llist set*
  assume
    R: $\forall x \in F.\ x \leq t \Longrightarrow \exists\ m \in F.\ \forall\ x \in F.\ x \leq m$
    and *t-upper-bound*: $\forall x \in$ *insert* x $F.\ x \leq t$
  then obtain m **where**
    m-in-F: $m \in F$ **and** *m-le-t*: $m \leq t$ **and** *x-le-t*: $x \leq t$
    and *m-max-F*: $\forall x \in F.\ x \leq m$ **using** R **by** (*auto dest*: R)
  from *m-le-t x-le-t* **have** $m \leq x \vee x \leq m$ **by** (*rule pref-locally-linear*)
  thus $\exists m \in$ *insert* x $F.\ \forall\ x \in$ *insert* x $F.\ x \leq m$
  proof
    assume $m \leq x$ **with** *m-max-F*
    have $\forall\ xa \in$ *insert* x $F.\ xa \leq x$ **by** *auto*
    thus *?thesis* **by** *blast*
  next assume $x \leq m$ **with** *m-max-F*
    have $\forall\ xa \in$ *insert* x $F.\ xa \leq m$ **by** *auto*
    thus *?thesis* **using** *m-in-F* **by** *blast*
  qed
qed

theorem *proposition-1-if*:
  assumes *S-Prop*: $S \in$ *Prop* **and** *lift-S-shp*: $[S] \in$ *SHP*
  shows $S \in$ *SP*
proof −
  { — Show that t has finite bad thing m.
    fix t :: *trace*
    assume *t-st*: $t \notin S\ \{t\} \in$ *Prop*
    then have *t-out-lift-S*: $\{t\} \notin [S]$ **by** (*simp add*: *property-lift-def*)

    obtain M **where**
      M-st: $M \in$ *Obs* $M \leq \{t\}\ \forall\ T'.\ T' \in$ *Prop* $\wedge\ M \leq T' \longrightarrow T' \notin [S]$
      using *t-out-lift-S* **and** *t-st* **and** *S-Prop* **and** *lift-S-shp*
      unfolding *SHP-def shp-def*
      by *blast*

    have $\exists\ ms \in \Psi_{\mathrm{fin}}.\ ms \in M\ \wedge\ ms \leq t\ \wedge\ (\forall\ m \in M.\ m \leq ms)$

```
proof −
  have M-pfx-t: ∀ m ∈ M. m ≤ t
    using M-st unfolding trace-set-prefix-def by blast
  have M-nonempty: M ≠ ∅
  proof (rule ccontr)
    {
      assume M-empty: ¬ M ≠ ∅
      {
        fix T' :: property assume T' ∈ Prop
        with M-empty have M ≤ T' unfolding trace-set-prefix-def by blast
      }
      hence M-pfx-Prop: ∀ T' ∈ Prop. M ≤ T' by blast
      have ∅ ∈ Prop unfolding Prop-def by blast
      hence M ≤ ∅ using M-pfx-Prop by blast
      hence ∅ ∉ [S] using M-st and ‹∅ ∈ Prop› by blast
      have ∅ ∈ [S] using property-lift-def by blast
      from ‹∅ ∈ [S]› and ‹¬ ∅ ∈ [S]› have False by blast
    }
    thus ¬ M ≠ ∅ ⟹ False by blast
  qed


  have M-fin: finite M using M-st unfolding Obs-def by blast
  from this obtain ms where ms-st: ms ∈ M ∀ x ∈ M. x ≤ ms
    using M-pfx-t and M-nonempty
    apply (insert prefix-set-has-longest [where t=t and X=M], blast)
    done
  hence ms-psi-fin: ms ∈ Ψfin using M-st unfolding Obs-def by blast
 have ms-pfx-t: ms ≤ t using ms-st and M-st unfolding trace-set-prefix-def
    by blast
  from ms-psi-fin and ms-st and ms-pfx-t
  show ∃ ms ∈ Ψfin. ms ∈ M ∧ ms ≤ t ∧ (∀ m ∈ M. m ≤ ms)
    by blast
qed
from this obtain m-star where
  m-star-st: m-star ∈ Ψfin  m-star ∈ M m-star ≤ t
           ∀ m ∈ M. m ≤ m-star
  by auto

{
  fix t'
  assume t'-st: {t'} ∈ Prop m-star ≤ t'
  let ?T' = {t'}
  have M ≤ ?T'
    proof −
```

{
fix m
assume $m \in M$
with *m-star-st* **have** $m \leq$ *m-star* **by** *blast*
with t'*-st* **have** $m \leq t'$ **using** *llist-le-trans* **by** *blast*
}
thus $M \leq$ *?T'* **unfolding** *trace-set-prefix-def* **by** *blast*
qed

with *M-st* **and** t'*-st* **have** *?T'* $\notin [S]$ **by** *blast*
hence $t' \notin S$ **unfolding** *property-lift-def* **by** *blast*
}

with *m-star-st* **have** $\exists\ m \in \Psi_{\text{fin}}.\ m \leq t \wedge (\forall\ t' \in \Psi_{\text{inf}}.\ m \leq t' \longrightarrow t' \notin S)$
unfolding *Prop-def*
by *blast*
}
thus *?thesis*
using *S-Prop*
unfolding *SP-def sp-def Prop-def* **by** *blast*
qed

4 Proposition 2

theorem *proposition-2-oif*:
fixes L :: *trace set*
assumes *L-Prop*: $L \in$ *Prop* **and** *L-LP*: $L \in$ *LP*
shows $[L] \in$ *LHP*
proof –
have *lift-L-HP*: $[L] \in$ *HP*
using *L-Prop property-lifts-into-hyperproperty* **by** *blast*
{
fix M **assume** *M-st*: $M \in$ *Obs*
{
fix m **assume** *m-st*: $m \in M$
have $\exists\ t.\ m \leq t \wedge t \in L$
proof –
from *m-st* **and** *M-st* **have** $m \in \Psi_{\text{fin}}$
unfolding *Obs-def* **by** *blast*
with *L-Prop* **and** *L-LP* **and** *m-st* **show** *?thesis*
unfolding *LP-def lp-def Prop-def* **by** *blast*
qed
}
hence *M-more*: $\forall\ m \in M.\ (\exists\ t.\ m \leq t \wedge t \in L)$ **by** *blast*

let $?T = \{t_m.\ \exists\ m \in M.\ m \leq t_m \wedge t_m \in L\}$
have $?T \subseteq L$ **by** *blast*
hence *T-in-lift*: $?T \in [L]$ **unfolding** *property-lift-def* **by** *blast*
with *M-more* **have** *M-pfx-T*: $M \leq ?T$
unfolding *trace-set-prefix-def* **by** *blast*
have $?T \in Prop$ **using** *M-st L-Prop*
unfolding *Prop-def psi-inf-def Obs-def psi-fin-def*
by *blast*
with *T-in-lift* **and** *M-pfx-T* **and** *L-Prop*
have $\exists\ T.\ T \in Prop \wedge M \leq T \wedge T \in [L]$ **by** *blast*
}
thus $[L] \in LHP$ **using** *lift-L-HP* **unfolding** *LHP-def lhp-def* **by** *blast*
qed

theorem *proposition-2-if*:
fixes L :: *trace set*
assumes *L-Prop*: $L \in Prop$ **and** *L-lift-lhp*: $[L] \in LHP$
shows $L \in LP$
proof $-$
{ fix t :: *trace* **assume** *t-st*: $t \in \Psi_{\text{fin}}$
let $?T = \{t\}$
obtain T' **where** T'*-st*: $?T \leq T' \quad T' \in [L] \quad T' \in Prop$
proof $-$
from *t-st* **have** *t-Obs*: $\{t\} \in Obs$ **using** *Obs-def* **by** *blast*
hence $\exists\ T' \in Prop.\ ?T \leq T' \wedge T' \in [L]$
using *L-lift-lhp* **unfolding** *LHP-def lhp-def* **by** *blast*
thus *?thesis* **by** *auto*
qed
then obtain t' **where** t'*-st*: $t \leq t' \quad t' \in T' \quad t' \in \Psi_{\text{inf}}$
unfolding *trace-set-prefix-def Prop-def* **by** *blast*
have $t' \in L$ **using** $\langle t' \in T' \rangle$ **and** $\langle T' \in [L] \rangle$
unfolding *property-lift-def* **by** *blast*
with t'*-st* **have** $\exists\ t' \in \Psi_{\text{inf}}.\ t \leq t' \wedge t' \in L$ **by** *blast*
}
thus $L \in LP$ **unfolding** *LP-def lp-def* **using** *L-Prop* **by** *blast*
qed

5 Theorem 3

5.1 Definitions and Lemmas

constdefs
Safe :: *hyperproperty* $\Rightarrow$ *hyperproperty*

Safe P $\triangleq$ $\{T \in$ *Prop*. $(\forall\ M \in$ *Obs*. $M \leq T \longrightarrow$
$(\exists\ T' \in$ *Prop*. $M \leq T' \wedge T' \in P))\}$
Live :: *hyperproperty* $\Rightarrow$ *hyperproperty*
Live P $\triangleq$ $P \cup$ (*Prop* $-$ *Safe P*)

lemma *Safe-is-HP*:
fixes P :: *hyperproperty*
assumes $P \in$ *HP*
shows *Safe P* $\in$ *HP*
unfolding *Safe-def HP-def* **by** *blast*

lemma *Live-is-HP*:
fixes P :: *hyperproperty*
assumes *P-HP*: $P \in$ *HP*
shows *Live P* $\in$ *HP*
using *P-HP*
unfolding *Live-def HP-def* **by** *blast*

lemma *Safe-is-hypersafety*:
fixes P :: *hyperproperty*
assumes *P-HP*: $P \in$ *HP*
shows *Safe P* $\in$ *SHP*
using *P-HP Safe-is-HP*
unfolding *Safe-def SHP-def shp-def*
by *blast*

lemma *P-subset-Safe-P*:
fixes P :: *hyperproperty*
assumes *P-HP*: $P \in$ *HP*
shows $P \subseteq$ *Safe P*
using *P-HP*
unfolding *Safe-def HP-def*
by *blast*

lemma *stutter-append-is-infinite*:
fixes x :: *trace*
assumes *x-fin*: $x \in \Psi_{\text{fin}}$ **and** *s-st*: $s \in \Sigma$
shows (x @@ *lconst s*) $\in \Psi_{\text{inf}}$
proof $-$
from *s-st* **have** *lconst s* $\in$ *inflsts* Σ
by (*rule lconstT* [*of s* Σ])
thus (x @@ *lconst s*) $\in \Psi_{\text{inf}}$
using *x-fin s-st lapp-fin-infT*
unfolding *psi-fin-def psi-inf-def*
by *blast*

qed

constdefs
asInfinite :: *trace* $\Rightarrow$ *trace*
asInfinite t $\triangleq$ *if LNil* = *t then lconst DummyState else t* @@ (*lconst* (*llast t*))
— Converts a finite trace to an infinite trace. If the given finite trace is non-empty, it returns a suffix in which the final state is infinitely stuttered; otherwise it returns the constant *DummyState* trace.

lemma *llast-in-trace-alphabet*:
assumes $t \in \Psi_{\text{fin}}$
shows $t \neq LNil \longrightarrow llast\ t \in \Sigma$ (**is** *?P t*)
using *prems*
unfolding *psi-fin-def*
by (*induct t rule*: *finlsts.induct*) *auto*

lemma *asInfinite-correctness*:
assumes *t-fin*: $t \in \Psi_{\text{fin}}$
shows *asInfinite* $t \in \Psi_{\text{inf}} \wedge t \leq$ *asInfinite t*
proof *cases*
assume *LNil* = *t*
thus *?thesis* **unfolding** *asInfinite-def psi-inf-def* **using** *DummyState-is-State*
by (*simp add*: *lconstT* [*of DummyState* Σ])
next
assume *t-positive*: *LNil* $\neq$ *t*
with *t-fin* **have** *res-inf*: *asInfinite* $t \in \Psi_{\text{inf}}$
proof−
have *llast* $t \in \Sigma$ **using** *t-positive t-fin llast-in-trace-alphabet* **by** *simp*
moreover
have *lconst* (*llast t*) $\in \Psi_{\text{inf}}$
using *t-fin t-positive* ‹*llast* $t \in \Sigma$› **unfolding** *psi-fin-def psi-inf-def*
by (*simp add*: *lconstT* [*of llast t* Σ])
moreover
have *t*@@*lconst* (*llast t*) $\in \Psi_{\text{inf}}$
using *t-fin* ‹*llast* $t \in \Sigma$›
by (*simp add*: *stutter-append-is-infinite* [*of t llast t*])
ultimately
show *asInfinite* $t \in \Psi_{\text{inf}}$ **unfolding** *asInfinite-def*
using *t-positive* **by** *simp*
qed
from *t-fin* **and** *t-positive*
have $t \leq$ *asInfinite t*
unfolding *psi-fin-def asInfinite-def* **using** *le-lappend* **by** *simp*
with *res-inf* **show** *?thesis* **..**

qed

lemma *Live-is-hyperliveness*:
  fixes P::*hyperproperty*
  assumes *P-HP*: $P \in HP$
  shows *Live* $P \in LHP$
proof −
  have *Live-HP*: *Live* $P \in HP$ **using** *P-HP Live-is-HP* **by** *blast*
  {
    fix T **assume** *T-st*: $T \in Obs$
    have $\exists\ T' \in Prop.\ T \leq T' \wedge T' \in Live\ P$
    proof *cases*
      assume $\exists\ T' \in Prop.\ T \leq T' \wedge T' \in P$
      then obtain T' **where** T'*-st*: $T' \in Prop\ T \leq T'\ T' \in P$ **by** *blast*
      hence $T' \in Live\ P$ **unfolding** *Live-def* **by** *blast*
      thus *?thesis* **using** T'*-st* **by** *blast*
    next
      assume T'*-non-extends*: $\neg(\exists\ T' \in Prop.\ T \leq T' \wedge T' \in P)$
      {
        fix T' **assume** T'*-extends-T*: $T' \in Prop\ T \leq T'$
        hence $T' \notin P$ **using** T'*-non-extends* **by** *blast*
        hence $T' \notin Safe\ P$
          proof −
            have $\exists\ T \in Obs.\ T \leq T' \wedge (\forall\ T' \in Prop.\ \neg(T \leq T') \mid (T' \notin P))$
              using *T-st* **and** T'*-extends-T* **and** T'*-non-extends* **by** *blast*
            hence $\neg(\forall\ M \in Obs.\ M \leq T' \longrightarrow$
                        $(\exists\ T'' \in Prop.\ M \leq T'' \wedge T'' \in P))$
              by *blast*
            thus *?thesis* **using** $\langle T' \in Prop \rangle$ **unfolding** *Safe-def* **by** *blast*
          qed
        hence $T' \in (Prop - Safe\ P)$ **using** $\langle T' \in Prop \rangle$ **by** *blast*
      }
      hence *all-pfx*: $\forall\ T' \in Prop.\ T \leq T' \longrightarrow T' \in Prop - Safe\ P$ **by** *simp*
      show $\exists\ T' \in Prop.\ T \leq T' \wedge T' \in Live\ P$
      proof −
        let $?T' = \{asInfinite\ x \mid x.\ x \in T\}$
        have T'*-suff*: $T \leq ?T'$ **using** *asInfinite-correctness T-st*
          unfolding *trace-set-prefix-def Obs-def* **by** *blast*
        have T'*-Prop*: $?T' \in Prop$ **using** *T-st asInfinite-correctness*
          unfolding *Obs-def Prop-def* **by** *blast*
        from T'*-suff* **and** T'*-Prop* **have** $?T' \in Prop - Safe\ P$ **using** *all-pfx* **by**
blast
        with T'*-suff* **and** T'*-Prop* **show** *?thesis* **unfolding** *Live-def* **by** *blast*
      qed

qed
}
thus *?thesis* **using** *Live-HP* **unfolding** *Live-def LHP-def lhp-def* **by** *blast*
qed

5.2 Theorem

theorem *theorem-3*:
fixes P :: *trace set set*
assumes *P-HP*: $P \in HP$
shows $\exists\ S \in SHP.\ \exists\ L \in LHP.\ P = S \cap L$
proof −
let $?S = Safe\ P$ **let** $?L = Live\ P$
have $?S \cap ?L = (P \cup Safe\ P) \cap (P \cup (Prop - Safe\ P))$
unfolding *Live-def* **using** *P-HP P-subset-Safe-P* **by** *blast*
also have $(P \cup Safe\ P) \cap (P \cup (Prop - Safe\ P))$
$= P \cap (Safe\ P \cup (Prop - Safe\ P))$
using *P-HP* **unfolding** *HP-def* **by** *blast*
also have $P \cap (Safe\ P \cup (Prop - Safe\ P)) = P \cap Prop$
unfolding *Safe-def* **by** *blast*
also have $P \cap Prop = P$ **using** *P-HP* **unfolding** *HP-def* **by** *blast*
finally have *witness*: $?S \cap ?L = P$ **by** *blast*

have *Safe-SHP*: $Safe\ P \in SHP$ **using** *Safe-is-hypersafety P-HP* **by** *blast*
have *Live-LHP*: $Live\ P \in LHP$ **using** *Live-is-hyperliveness P-HP* **by** *blast*

show *?thesis* **using** *Safe-SHP Live-LHP witness* **by** *blast*
qed

6 Theorem 1

6.1 Definitions and Lemmas

constdefs
Systems :: *trace set set*
$Systems \triangleq \{ts.\ ts \neq \emptyset \wedge ts \subseteq \Psi_{\mathrm{inf}}\}$
refinedby :: *trace set* ⇒ *trace set* ⇒ *bool* (**infix** $\leq$ *80*)
$S \leq S' \triangleq S' \subseteq S$
rc :: *hyperproperty* ⇒ *bool*
$rc\ H \triangleq \forall\ S \in Systems.\ S \models H \longrightarrow$
$(\forall\ S' \in Systems.\ S \leq S' \longrightarrow S' \models H)$
RC :: *hyperproperty set*
$RC \triangleq \{H \in HP.\ rc\ H\}$

axioms *safety-and-liveness-onlyif-true*:

$\llbracket\ p \in LP;\ p \in SP\ \rrbracket \Longrightarrow p = \textit{true-Prop}$
— Any property which is both safety and liveness is the *true* property. This is axiomatised since it is well-known about the theory of properties.

lemma *hypersafety-and-hyperliveness-onlyif-true*:
  fixes H :: *hyperproperty*
  assumes *H-SHP*: $H \in SHP$ **and** *H-LHP*: $H \in LHP$
  shows $H = \textit{true-HP}$
proof (*rule ccontr*)
  have *H-HP*: $H \in HP$ **using** *H-SHP* **unfolding** *SHP-def shp-def* **by** *blast*
  {
    assume *H-untrue*: $H \neq \textit{true-HP}$
    then obtain *Tstar* **where** *Tstar-st*: $Tstar \in Prop\ \ Tstar \notin H$
      using *H-HP* **unfolding** *HP-def true-HP-def Prop-def* **by** *blast*
    obtain M **where** *M-st*: $M \in Obs\ \ M \leq Tstar$
      $\forall\ T' \in Prop.\ M \leq T' \longrightarrow T' \notin H$
      using *H-SHP Tstar-st*
      unfolding *SHP-def shp-def* **by** *blast*
    then obtain *Th* **where** *Th-st*: $Th \in Prop\ \ M \leq Th\ \ Th \in H$
      using *H-LHP*
      unfolding *LHP-def lhp-def* **by** *blast*
    hence $Th \notin H$ **using** ‹$Th \in Prop$› *M-st* **by** *blast*
    thus *False* **using** *Th-st* **by** *blast*
  }
qed

lemma *hypersafety-and-hyperliveness-onlyif-true-contrapos*:
  fixes H :: *hyperproperty*
  shows $H \neq \textit{true-HP} \longrightarrow (H \notin LHP \mid H \notin SHP)$
  apply (*insert hypersafety-and-hyperliveness-onlyif-true* [*of H*])
  by *blast*

axioms *Ex-nontrue-Prop*: $\exists\ l \in LP.\ l \neq \textit{true-Prop}$
— There is a liveness property other than *true*. This is axiomatised since it is well-known about the theory of properties.

lemma *system-is-property*:
  fixes s :: *trace set*
  assumes *s-Sys*: $s \in Systems$
  shows $s \in Prop$
using *s-Sys*
unfolding *Systems-def Prop-def* **by** *blast*

lemma *HP-contains-SHP*: $SHP \subseteq HP$ **unfolding** *SHP-def shp-def* **by** *blast*

6.2 Theorem

theorem *theorem-1-relaxed*:
  shows $SHP \subseteq RC$
proof (*rule ccontr*)
  assume $\neg$ $SHP \subseteq RC$
  then obtain S **where** *S-SHP*: $S \in SHP$ **and** *S-not-RC*: $S \notin RC$ **by** *blast*
  have *S-HP*: $S \in HP$ **using** *S-SHP HP-contains-SHP* **by** *blast*
  from *S-HP* **and** *S-not-RC*
  obtain T T' **where** *T-st*: $T \in Prop$ $T \in S$
    and *T'-st*: $T' \in Prop$ $T' \notin S$
    and *T-gt-T'*: $T \supseteq T'$
    unfolding *RC-def rc-def HP-def Systems-def Prop-def*
    unfolding *refinedby-def hyperproperty-satisfies-def*
    by *blast*
  from *T'-st* **obtain** M
    where *M-st*: $M \leq T'$ $(\forall\ T'' \in Prop.\ M \leq T'' \longrightarrow T'' \notin S)$
    using *S-SHP* **unfolding** *SHP-def shp-def* **by** *blast*
  have $M \leq T$
    using *M-st T-st T'-st T-gt-T'*
    unfolding *trace-set-prefix-def* **by** *blast*
  hence $T \notin S$ **using** *T-st M-st* **by** *blast*
  thus *False* **using** *T-st* **by** *blast*
qed

theorem *theorem-1*: $SHP \subset RC$
proof
  show $SHP \subseteq RC$ **using** *theorem-1-relaxed* **by** *assumption*
  obtain l **where** *l-LP*: $l \in LP$ **and** *l-untrue*: $l \neq$ *true-Prop*
    using *Ex-nontrue-Prop* **by** *blast*
  hence *cx-RC*: $[l] \in RC$
    unfolding *property-lift-def LP-def lp-def RC-def rc-def Systems-def*
      refinedby-def HP-def Prop-def psi-inf-def psi-fin-def
      hyperproperty-satisfies-def
    by *blast*
  from *l-untrue* **have** $[l] \neq$ *true-HP*
    using *l-LP*
    unfolding *LP-def lp-def true-Prop-def true-HP-def property-lift-def*
      psi-inf-def Prop-def
    by *blast*
  hence $[l] \notin SHP$
  proof −
    have $l \in Prop$ **using** *l-LP* **unfolding** *LP-def lp-def* **by** *blast*
    with *l-LP* **have** $[l] \in LHP$ **using** *proposition-2-oif* **by** *blast*
    thus $[l] \notin SHP$

using $\langle [l] \neq \textit{true-HP} \rangle$
hypersafety-and-hyperliveness-onlyif-true-contrapos **by** *blast*
qed
thus $SHP \neq RC$ **using** *cx-RC* **by** *blast*
qed

7 Proposition 3

7.1 Definitions and Lemmas

constdefs
$Cls :: ('a\ set \Rightarrow 'a\ set)\ set$
$Cls \triangleq \{cl.\ \forall\ T :: 'a\ set.\ T \subseteq cl\ T\}$

$PIF :: hyperproperty\ set$
$PIF \triangleq \{\{Cl\ T \mid T.\ T \in Prop\} \mid Cl.\ Cl \in Cls\}$

$lsingle :: 'a \Rightarrow 'a\ llist$
$lsingle\ x \triangleq x\#\#LNil$

$hasDummyState :: trace \Rightarrow bool$
$hasDummyState\ t \triangleq \exists\ t'.\ t'@@(lsingle\ DummyState) \leq t$

$GS :: trace\ set$
$GS \triangleq \{t.\ t \in \Psi_{\mathrm{inf}} \wedge hasDummyState\ t\}$
— The guaranteed service property, *GS*, contains infinite traces in which a designated state occurs. This definition generalizes *GS* from the technical report.

axioms
Cl-produces-Props: $[\![\ T \in Prop;\ Cl \in Cls\ ]\!] \Longrightarrow Cl\ T \in Prop$
— This axiom is essentially a type signature on closures. It is axiomatised because although it is not mentioned in the technical report, it is required for Proposition 3.

EX-trace-sans-DummyState: $\exists\ t \in \Psi_{\mathrm{inf}}.\ \neg hasDummyState\ t$
— There is an infinite trace without a certain state (the *DummyState*, in this case). This is axiomatised because it is well-known about the theory of properties.

GS-liveness: $lp\ GS$
— The *GS* property is a liveness property. This is axiomatised since it is well-known.

lemma *GS-LHP*: $[GS] \in LHP$
proof $-$
  have $GS \in Prop$ **unfolding** *Prop-def GS-def* **by** *blast*
  thus *?thesis* **using** *GS-liveness proposition-2-oif* **unfolding** *LP-def* **by** *blast*
qed

lemma *trace-set-prefix-expanding'*:
  fixes T :: *trace set*
  assumes *T-st*: $T \le T'$ **and** *T'-sub*: $T' \subseteq T''$
  shows $T \le T''$
  using *T-st T'-sub* **unfolding** *trace-set-prefix-def* **by** *blast*

7.2 Proposition

theorem *proposition-3-relaxed*:
  shows $PIF \subseteq LHP$
proof $-$
  {
    fix P **assume** $P \in PIF$
    then obtain *Cl-P* **where** *P-st*: $P = \{Cl\text{-}P\ T \mid T.\ T \in Prop\}$
                  and *Cl-P-closure*: $Cl\text{-}P \in Cls$
      unfolding *PIF-def* **by** *blast*
    have *P-HP*: $P \in HP$
    proof $-$
      {
        fix x **assume** $x \in P$
        then obtain T **where** *T-st*: $x = Cl\text{-}P\ T\ T \in Prop$
          using *P-st* **by** *blast*
        hence $x \in Prop$ **using** *Cl-P-closure Cl-produces-Props* **by** *blast*
      }
      thus *?thesis* **unfolding** *HP-def* **by** *blast*
    qed
    {
      fix T **assume** *T-Obs*: $T \in Obs$
      have $\exists\ T' \in Prop.\ T \le T' \wedge T' \in P$
      proof $-$
        let $?T\text{-}inf = \{asInfinite\ t \mid t.\ t \in T\}$
        let $?T' = Cl\text{-}P\ ?T\text{-}inf$
        have *T'-suff*: $T \le ?T'$
        proof $-$
          have *Cl-P-monotonic*: $\bigwedge X.\ X \subseteq Cl\text{-}P\ X$
            using *Cl-P-closure* **unfolding** *Cls-def* **by** *blast*
          hence *Cl-P-prop*: $?T\text{-}inf \subseteq Cl\text{-}P\ ?T\text{-}inf$ **by** *auto*
          have *T-pfx-T-inf*: $T \le ?T\text{-}inf$
            using *T-Obs asInfinite-correctness*

unfolding *Obs-def trace-set-prefix-def* **by** *blast*
with *Cl-P-prop* **show** *?thesis*
apply (*insert trace-set-prefix-expanding′* [*OF T-pfx-T-inf Cl-P-prop*])
apply *assumption*
done
qed
have *?T-inf* $\in$ *Prop* **using** *T-Obs asInfinite-correctness*
unfolding *Obs-def Prop-def* **by** *blast*
hence *T′-P*: *?T′* $\in$ *P* **using** *P-st* **by** *blast*
have *T′-Prop*: *?T′* $\in$ *Prop*
using ‹*?T-inf* $\in$ *Prop*› *Cl-P-closure Cl-produces-Props* **by** *blast*
with ‹*?T′* $\in$ *P*› **and** *T′-suff* **show** *?thesis* **by** *blast* **qed**
}
hence *P* $\in$ *LHP* **using** *P-HP* **unfolding** *LHP-def lhp-def* **by** *blast*
}
thus *PIF* $\subseteq$ *LHP* **by** *blast*
qed

theorem *proposition-3*:
shows *PIF* $\subset$ *LHP*
proof
show *PIF* $\subseteq$ *LHP* **using** *proposition-3-relaxed* **.**
have *GS-lift-LHP*: [*GS*] $\in$ *LHP* **by** (*simp add*: *GS-LHP*)
show *PIF* $\neq$ *LHP*
proof (*rule ccontr*)
{
assume *PIF* = *LHP*
hence [*GS*] $\in$ *PIF* **using** *GS-lift-LHP* **by** *simp*
then obtain *CL-GS*
where *CL-GS-st*: [*GS*] = {*CL-GS T* | *T*. *T* $\in$ *Prop*}
and *CL-GS-Cls*: *CL-GS* $\in$ *Cls* **unfolding** *PIF-def* **by** *blast*
obtain *t*
where *t-inftrace*: *t* $\in \Psi_{\text{inf}}$
and *t-no-Dummy*: $\neg$ *hasDummyState t*
using *EX-trace-sans-DummyState* **by** *blast*
hence *ts-Prop*: {*t*} $\in$ *Prop* **unfolding** *Prop-def* **by** *blast*
have *t* $\in$ *CL-GS* {*t*} **using** *CL-GS-Cls* **unfolding** *Cls-def* **by** *blast*
hence $\neg$ (*CL-GS* {*t*} $\models$ *GS*)
using *t-no-Dummy*
unfolding *property-satisfies-def GS-def* **by** *blast*
hence *False* **using** *CL-GS-st*
using *ts-Prop* **unfolding** *property-satisfies-def property-lift-def* **by** *blast*
}
thus $\neg$ *PIF* $\neq$ *LHP* $\Longrightarrow$ *False* **by** *blast*
qed

qed

8 Theorem 2

8.1 Definitions and Lemmas

We represent traces over the alphabet A^k as $'a$ *llist llist* where $'a$ is the type of elements of A. That is, instead of using k-tuples, we use llists of length k.

constdefs

$kshp :: nat \Rightarrow hyperproperty \Rightarrow bool$
$kshp\ k\ S \triangleq$
$\quad S \in HP\ \wedge$
$\quad (\forall\ T \in Prop.\ T \notin S \longrightarrow$
$\qquad (\exists\ M \in Obs.\ M \leq T \wedge card\ M = k\ \wedge$
$\qquad\quad (\forall\ T' \in Prop.\ M \leq T' \longrightarrow T' \notin S)))$
$KSHP :: nat \Rightarrow hyperproperty\ set$
$KSHP\ k \triangleq \{S.\ kshp\ k\ S\}$

$fromSome :: 'a\ option \Rightarrow 'a$
$fromSome\ x \triangleq (case\ x\ of\ Some\ e \Rightarrow e \mid None \Rightarrow arbitrary)$
$fromSomeSt :: state\ option \Rightarrow state$
$fromSomeSt\ x \triangleq (case\ x\ of\ Some\ s \Rightarrow s \mid None \Rightarrow \bot)$

$zipn :: nat \Rightarrow (state\ llist)\ llist \Rightarrow (state\ llist)\ llist \Rightarrow bool$
$zipn\ k\ T\ t \triangleq$
$\quad \forall\ j :: nat.\ j < k \longrightarrow t!!j = Some\ (lmap\ (\lambda t.\ fromSomeSt\ (t!!j))\ T)$
— The zip relation. We get unzip for free.

$set\text{-}to\text{-}llist :: 'a\ set \Rightarrow 'a\ llist$
$set\text{-}to\text{-}llist\ S \triangleq SOME\ l.\ lset\ l = S$

Following are various axioms about the *zip* operator. Each axiom corresponds to an unproved fact about the operator.

axioms

zip-of-Obs-exists:
$M \in Obs \Longrightarrow \exists\ m.\ zipn\ k\ (set\text{-}to\text{-}llist\ M)\ m$
— Any observation can be zipped. This axiom is used in the if direction of theorem 3.

zip-EX-suffix:
$[\![\ M \in Obs;\ S \in Systems;\ zipn\ k\ (set\text{-}to\text{-}llist\ M)\ m;\ M \leq S\ ]\!]$
$\Longrightarrow \exists\ s \in kProd\ k\ S.\ prefix\text{-}k\ k\ m\ s$
— There is a suffix S^k to any zip of an observation, if the system S is a suffix of the observation.

zip-of-Obs-fin:
$[\![$ $M \in$ *Obs*; *zipn* k (*set-to-llist* M) m $]\!]$
$\Longrightarrow m \in (\Sigma^\star)^\star$
— Zipping an observation produces a finite trace over Σ^k.

unzipped-recoverable:
zipn k (*set-to-llist* M) *Mz*
$\Longrightarrow \forall\ j{<}k.\ \exists\ m \in M.\ m =$ *lmap* ($\lambda t.$ *fromSome* ($t!!j$)) *Mz*
— Every member from an unzipped trace set corresponds to some element of the zip.

unzip-monotonic-wrt-prefix-k:
$[\![$ *zipn* k (*set-to-llist* M) *Mz*; *zipn* k (*set-to-llist* T) *Tz*; *prefix-k* k *Mz Tz* $]\!]$
$\Longrightarrow M \leq$ *Tl*
— Unzipping is monotonic.

constdefs
noBot :: *state llist* $\Rightarrow$ *bool*
noBot $\triangleq$ *finlsts-rec True* ($\lambda\ s\ r\ b.\ b \wedge (s \neq \bot)$)
— *noBot* t asserts that the finite trace t does not contain $\bot$.

bottoms :: *state llist* — infinite list of bottoms
bottoms $\triangleq$ *lconst* $\bot$

prefix-bottom :: *state llist* $\Rightarrow$ *state llist* $\Rightarrow$ *bool* (**infix** $\leq_\bot$ *60*)
$t \leq_\bot u \triangleq \exists\ tp.$ *noBot* $tp \wedge t \leq tp$ @@ *bottoms* $\wedge\ tp \leq u$
— Effectively removes the bottoms from the first trace, then compares it to the second.

prefix-k :: (*state llist*) *llist* $\Rightarrow$ *nat* $\Rightarrow$ (*state llist*) *llist* $\Rightarrow$ *bool* (- $\leq_{-}$ - *60*)
$t_k \leq_k u_k \triangleq$
$\forall\ j.\ j < k \longrightarrow$
(*lmap* ($\lambda t.$ *fromSome* ($t!!j$)) t_k) $\leq_\bot$ (*lmap* ($\lambda t.$ *fromSome* ($t!!j$)) u_k)
— The input traces are over the alphabet Σ^k. We project the jth position of each element, which creates two traces each with state elements, and compare those with *prefix-bottom*.

State-K :: *state llist set*
State-K $\triangleq \Sigma^\star$
TraceFin-K :: *state llist llist set*
TraceFin-K $\triangleq$ *State-K**
TraceInf-K :: *state llist llist set*
TraceInf-K $\triangleq$ *State-K*$^\omega$
Prop-K :: *state llist llist set set*

$Prop\text{-}K \triangleq Pow\ TraceInf\text{-}K$

A generic definition of safety which takes an alphabet as a parameter. For theorem 2 we require reasoning about traces over Σ and Σ^k.

constdefs

$spa :: nat \Rightarrow (state\ llist)\ llist\ set \Rightarrow bool$
$spa\ k\ P \triangleq P \in Prop\text{-}K$
$\quad \wedge (\forall\ t \in TraceInf\text{-}K.\ t \notin P \longrightarrow$
$\quad\quad (\exists\ m \in TraceFin\text{-}K.\ m \leq_k t \wedge$
$\quad\quad\quad (\forall\ t' \in TraceInf\text{-}K.\ m \leq_k t' \longrightarrow t' \notin P)))$

$SPA :: nat \Rightarrow (state\ llist)\ llist\ set\ set$
$SPA\ k \triangleq \{P.\ spa\ k\ P\}$

$kProd :: nat \Rightarrow state\ llist\ set \Rightarrow (state\ llist)\ llist\ set$
$kProd\ k\ S \triangleq \{t \in TraceInf\text{-}K.\ \exists\ S' \in Systems.$
$\quad S' \subseteq S \wedge card\ S' = k \wedge zipn\ k\ (set\text{-}to\text{-}llist\ S')\ t\}$
— k-product of a system S.

$pa\text{-}satisfies :: 'a\ llist\ set \Rightarrow 'a\ llist\ set \Rightarrow bool\ ((\text{-} \models \text{-})\ [80,80]\ 80)$
$pa\text{-}satisfies\text{-}def: ts \models p \triangleq ts \subseteq p$
— Whether a set of traces over an alphabet $'a$ satisfies a property.

$KSP :: nat \Rightarrow (state\ llist)\ llist\ set\ set$
$KSP\ k \triangleq SPA\ k$

$Bads\text{-}from\text{-}KSaf :: nat \Rightarrow hyperproperty \Rightarrow trace\ set\ set$
$Bads\text{-}from\text{-}KSaf\ k\ KK \triangleq$
$\{M \in Obs.\ card\ M \leq k$
$\quad \wedge (\exists\ T \in Prop.\ T \notin KK \wedge M \leq T)$
$\quad \wedge (\forall\ T' \in Prop.\ M \leq T' \longrightarrow T' \notin KK)\ \}$
— Boldface M in the proof of theorem 2.

$Saf\text{-}from\text{-}KSaf :: nat \Rightarrow hyperproperty \Rightarrow (state\ llist)\ llist\ set$
$Saf\text{-}from\text{-}KSaf\ k\ KK \triangleq$
$\{t \in TraceInf\text{-}K.$
$\quad \neg(\exists\ M \in Obs.\ \exists\ tz \in TraceFin\text{-}K.$
$\quad\quad M \in Bads\text{-}from\text{-}KSaf\ k\ KK \wedge zipn\ k\ (set\text{-}to\text{-}llist\ M)\ tz \wedge tz \leq_k t)\}$
— Boldface K in the proof of theorem 2.

lemma *Saf-from-KSaf-is-safety*:
fixes $k :: nat$
assumes *KK-KSHP*: $KK \in KSHP\ k$
shows $Saf\text{-}from\text{-}KSaf\ k\ KK \in KSP\ k$
proof –

```
  let ?K = Saf-from-KSaf k KK
  have Saf-from-KSaf-st: ?K ∈ Prop-K
    unfolding Saf-from-KSaf-def TraceInf-K-def State-K-def Prop-K-def
    by blast
  {
    fix t assume t-st: t ∈ TraceInf-K t ∉ ?K
    then have ∃M ∈ Obs. M ∈ Bads-from-KSaf k KK
                    ∧ (∃ tz ∈ TraceFin-K. zipn k (set-to-llist M) tz ∧ tz ≤_k t)
      unfolding Saf-from-KSaf-def TraceInf-K-def TraceFin-K-def State-K-def
      by blast
    then obtain M tz where M-tz-st:
                    M ∈ Obs
                    M ∈ Bads-from-KSaf k KK
                    tz ∈ TraceFin-K
                    zipn k (set-to-llist M) tz
                    tz ≤_k t
      by blast
    {
      fix u assume u-st: u ∈ TraceInf-K tz ≤_k u
      hence u ∉ ?K using M-tz-st
        unfolding TraceInf-K-def Saf-from-KSaf-def State-K-def by blast
    }
    hence ∃ tz ∈ TraceFin-K.
              tz ≤_k t ∧ (∀ u ∈ TraceInf-K. tz ≤_k u ⟶ u ∉ Saf-from-KSaf k KK)
      using M-tz-st unfolding TraceFin-K-def TraceInf-K-def State-K-def
      by blast
  }
  thus ?K ∈ KSP k
    unfolding KSP-def SPA-def spa-def using Saf-from-KSaf-st by blast
qed


lemma trace-set-prefix-transitive:
  assumes X-p-Y: X ≤ Y and Y-p-Z: Y ≤ Z
  shows X ≤ Z
proof−
  {
    fix x assume x ∈ X
    then obtain y where y ∈ Y  x ≤ y
      using X-p-Y unfolding trace-set-prefix-def by blast
    then obtain z where z ∈ Z  y ≤ z
      using Y-p-Z unfolding trace-set-prefix-def by blast
    have x ≤ z using ‹x ≤ y› ‹y ≤ z›
      by (rule llist-le-trans [of x y z])
    hence ∃ z ∈ Z. x ≤ z using ‹z ∈ Z› by blast
```

}
thus $X \leq Z$ **unfolding** *trace-set-prefix-def* **by** *blast*
qed

8.2 Theorem

theorem *theorem-2-onlyif*:
fixes k :: *nat*
assumes *S-Sys*: $S \in$ *Systems* **and** *KK-KSHP*: $KK \in$ *KSHP k*
shows $\exists\ K \in$ *KSP k*. $((S \models (KK$ :: *hyperproperty*$)) \longrightarrow (($*kProd k S*$) \models K))$
proof−
let *?K* = *Saf-from-KSaf k KK*
let *?MM* = *Bads-from-KSaf k KK*
let *?S-k* = *kProd k S*
have *K-is-safety*: *?K* $\in$ *KSP k* **using** *KK-KSHP* **by** (*simp add*: *Saf-from-KSaf-is-safety*)

have $(S \models (KK$ :: *hyperproperty*$)) \longrightarrow (($*?S-k*$) \models$ *?K*$)$
proof (*rule ccontr*)
{
assume *neg*: $\neg$ $(S \models (KK$ :: *hyperproperty*$) \longrightarrow ($*?S-k*$) \models$ *?K*$)$
hence *S-Sat-KK*: $S \models KK$ **by** *blast*
have *S-k-Unsat*: $\neg$ $(($*?S-k*$) \models$ *?K*$)$ **using** *neg* **by** *blast*
have *S-in-KK*: $S \in KK$
using *S-Sat-KK* **unfolding** *hyperproperty-satisfies-def* **.**
have *S-unsub-K*: $\neg$ *?S-k* $\subseteq$ *?K* **using** *S-k-Unsat* **unfolding** *pa-satisfies-def* **.**
then obtain t **where** *t-st*: $t \in$ *?S-k* $t \notin$ *?K* **by** *blast*
hence $t \in$ *TraceInf-K* **unfolding** *kProd-def* **by** *blast*
then obtain M *zip-M* **where** *M-zip-M-st*: $M \in$ *Obs*
$M \in$ *?MM*
zipn k (*set-to-llist M*) *zip-M*
zip-M $\leq_k$ t
using *t-st* **unfolding** *Saf-from-KSaf-def* **by** *blast*
obtain T **where** *T-st*: *zipn k* (*set-to-llist T*) t
$T \in$ *Prop*
$T \subseteq S$
using ‹$t \in$ *?S-k*› **unfolding** *kProd-def Systems-def Prop-def* **by** *blast*
have *M-pfx-T*: $M \leq T$ **using** ‹*zipn k* (*set-to-llist T*) t›
‹*zipn k* (*set-to-llist M*) *zip-M*›
‹*zip-M* $\leq_k$ t›
by (*simp add*: *unzip-monotonic-wrt-prefix-k*)
hence $T \notin KK$ **using** ‹$M \in$ *?MM*› ‹$T \in$ *Prop*›
unfolding *Bads-from-KSaf-def* **by** *blast*
have $T \leq S$ **using** *T-st S-Sys* ‹$t \in$ *?S-k*›
unfolding *trace-set-prefix-def Systems-def kProd-def zipn-def*
by *blast*

with *M-pfx-T* **have** *M-pfx-S*: $M \leq S$
by (*rule trace-set-prefix-transitive* [*of* M T S])

have $S \in$ *Prop* **using** *S-Sys* **unfolding** *Systems-def Prop-def* **by** *blast*
have $S \notin KK$ **using** *M-zip-M-st M-pfx-S* ‹$S \in$ *Prop*› **unfolding** *Bads-from-KSaf-def*
by *blast*
with *S-in-KK* **have** *False* **by** *simp*
}
thus $\neg$ ($S \models KK \longrightarrow$ *kProd* k $S \models$ *?K*) $\Longrightarrow$ *False* **by** *assumption*
qed
thus $\exists$ $K \in$ *KSP* k. $S \models KK \longrightarrow$ *kProd* k $S \models K$
using *K-is-safety* **by** *blast*
qed

theorem *theorem-2-if*:
fixes k :: *nat*
assumes *S-Sys*: $S \in$ *Systems* **and** *KK-KSHP*: $KK \in$ *KSHP* k
shows $\exists$ $K \in$ *KSP* k. (((*kProd* k S) $\models K$) $\longrightarrow$ ($S \models$ (KK :: *hyperproperty*)))
proof−
let *?K* = *Saf-from-KSaf* k KK
let *?M* = *Bads-from-KSaf* k KK
let *?S-k* = *kProd* k S
have *K-is-safety*: *?K* $\in$ *KSP* k **using** *KK-KSHP* **by** (*simp add*: *Saf-from-KSaf-is-safety*)
have ((*?S-k* $\models$ *?K*) $\longrightarrow$ ($S \models$ (KK :: *hyperproperty*)))
proof (*rule ccontr*)
{ **assume** *neg*: $\neg$ (((*?S-k*) $\models$ *?K*) $\longrightarrow$ ($S \models$ (KK :: *hyperproperty*)))
hence *?S-k* $\subseteq$ *?K* **unfolding** *pa-satisfies-def* **by** *simp*
have $\neg$ ($S \models KK$) **using** *neg* **by** *simp*
have $S \in$ *Prop* **using** *S-Sys* **unfolding** *Prop-def Systems-def* **by** *blast*
hence $S \notin KK$ **using** ‹$\neg$ ($S \models KK$)›
unfolding *hyperproperty-satisfies-def* **by** *simp*

hence
$\exists$ $M \in$ *Obs*. $M \leq S \wedge$ *card* $M = k \wedge$
($\forall$ $T' \in$ *Prop*. $M \leq T' \longrightarrow T' \notin KK$)
using ‹$S \in$ *Prop*› *KK-KSHP* **unfolding** *KSHP-def kshp-def*
by *blast*
then obtain M **where** *M-st*: $M \leq S$ *card* $M = k$ $M \in$ *Obs*
$\forall$ $T' \in$ *Prop*. $M \leq T' \longrightarrow T' \notin KK$ **by** *blast*
have $\exists$ m. *zipn* k (*set-to-llist* M) m
using ‹$M \in$ *Obs*› **by** (*simp add*: *zip-of-Obs-exists* [*of* M k])
then obtain m **where** *m-st*: *zipn* k (*set-to-llist* M) m **by** *blast*
obtain s **where** $s \in$ *?S-k* $m \leq_k s$
using ‹$M \in$ *Obs*› ‹$S \in$ *Systems*› *m-st* ‹$M \leq S$›
using *zip-EX-suffix* **by** *best*

have $M \in ?M$ **unfolding** *Bads-from-KSaf-def* **using** *M-st* ‹$S \in Prop$›
  by *blast*
have $m \in TraceFin\text{-}K$ **unfolding** *TraceFin-K-def*
  using *m-st* ‹$M \in Obs$› *zip-of-Obs-fin*
  unfolding *zipn-def State-K-def Obs-def psi-fin-def*
  by *blast*
have $s \notin ?K$ **unfolding** *Saf-from-KSaf-def*
  using ‹$M \in Obs$› ‹$m \in TraceFin\text{-}K$› ‹$M \in ?M$› ‹$zipn\ k\ (set\text{-}to\text{-}llist\ M)\ m$›
    ‹$m \leq_k s$› **by** *blast*
hence $\neg\ ?S\text{-}k \subseteq ?K$ **using** ‹$s \in ?S\text{-}k$› **by** *blast*
hence *False* **using** ‹$?S\text{-}k \subseteq ?K$› **by** *blast*
}
thus $\neg\ (kProd\ k\ S \models Saf\text{-}from\text{-}KSaf\ k\ KK \longrightarrow S \models KK) \Longrightarrow False$ **by** *assumption*
qed
thus $\exists\ K \in KSP\ k.\ kProd\ k\ S \models K \longrightarrow S \models KK$
  using *K-is-safety* **by** *blast*
qed

end

References

[1] Denis L. Bueno and Michael R. Clarkson. Hyperproperties: Verification of proofs. Cornell University Computing and Information Science Technical Report, http://hdl.handle.net/1813/11153, July 2008.

[2] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. Cornell University Computing and Information Science Technical Report, http://hdl.handle.net/1813/9480, January 2008.

[3] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. In *Proc. IEEE Computer Security Foundations Symposium*, pages 51–65, June 2008.

[4] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer-Verlag, London, Cambridge, Massachusetts, 2002.